# *Weighted Fuzzy Rules Optimised by Particle Swarm for Network Intrusion Detection*

Chen, Tianhua; Su, Pan; Shang, Changjing; Shen, Qiang

*Published in:*
2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)

*DOI:*
10.1109/FUZZ-IEEE.2018.8491553

*Publication date:*
2018

*Citation for published version (APA):*
Chen, T., Su, P., Shang, C., & Shen, Q. (2018). Weighted Fuzzy Rules Optimised by Particle Swarm for Network Intrusion Detection. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (International Conference on Fuzzy Systems). IEEE Press. https://doi.org/10.1109/FUZZ-IEEE.2018.8491553

# Weighted Fuzzy Rules Optimised by Particle Swarm for Network Intrusion Detection

Tianhua Chen[a], Pan Su[b], Changjing Shang[c] and Qiang Shen[c]

[a]Department of Computer Science, School of Computing and Engineering, University of Huddersfield, UK
[b]School of Control and Computer Engineering, North China Electric Power University, China
[c]Department of Computer Science, Institute of Mathematics, Physics and Computer Science, Aberystwyth University, UK

***Abstract*—Network intrusion detection systems (IDSs) dynamically monitor communication events on a network, and decide whether any event is symptomatic of an attack or constitutes a legitimate use of the system. They have become an indispensable component of security infrastructure, e.g., to detect threats before widespread damage takes place. A variety of approaches have been proposed to design IDSs, including fuzzy rule-based techniques that offer advantages such as tolerance of noisy and imprecise data. In particular, fuzzy rules can be highly interpretable and trackable if the underlying fuzzy sets are predefined, directly reflecting domain expertise. This paper proposes such an approach to generate a set of weighted fuzzy rules for building effective IDSs, where the rule weights are optimised by Particle Swarm Optimisation without affecting the underlying predefined fuzzy sets. Experiments are performed on benchmark IDS datasets with comparison to alternative systems built with popular machine learning methods.**

## I. Introduction

With the rapid development and explosive use of the Internet and computer systems, the amount of malicious intrusion or attack on computer networks has excessively increased year by year [1]. Indeed, techniques for ensuring cyber security is of paramount importance for any governmental, organisational or even personal use of the Internet [2]. As a complementary solution to firewall technology, network intrusion detection systems (IDSs) aim to identify unauthorised, illicit, and anomalous behaviour based on network traffic to support decision making for devising preventive actions by network administrators [3].

IDSs dynamically monitor any communication events on a computer network, and decide whether these events are symptomatic of an attack or constitute a legitimate use of the system. In general, there are two types of IDS, namely, signature-based IDSs and anomaly-based IDSs [1]. Signature-based detection systems aim to identify intrusions that match with previously known attacks. They are very efficient, but are limited to the information from which they were trained. Anomaly-based detection systems are introduced to address suspicious patterns whose behaviour deviates from established normal patterns. Such systems are able to detect attacks that have not previously experienced, but strongly depend on the continuity of monitoring the expected normal activity [4].

Independent of which type of IDSs, their development requires domain knowledge. Fuzzy rule induction forms a major approach to learning robust interpretable knowledge models. Indeed, many techniques (e.g., [5], [6], [7], [8], [9], [10]) have been proposed for learning fuzzy if-then rules from numerical data. Apart from being of fault tolerance and error resilience in the face of noisy and imprecise data, which fits the requirements of building effective IDSs, fuzzy models also allow for enhanced transparency in both the learned models themselves and the inferences performed with the resulting models [11]. However, a major challenge in learning fuzzy rules often exists where the membership functions defining the antecedent fuzzy sets are prefixed, with each having a predefined linguistic meaning by domain experts or users. The incorporation of intuitive expert knowledge into linguistic rules through the use of prespecified fuzzy sets is desirable to effectively interpret a fuzzy model [12]. Yet, using just a fixed quantity space consisting of such fuzzy sets inevitably limits the accuracy of the learnt rules.

Fortunately, this problem can be tackled by modifying the weights associated with the individual rules. Rule weights intuitively reveal the relative importance amongst all the rules in a given rule base. The greater the weight associated with a fuzzy if-then rule, the more likely it will be chosen for use (i.e., for firing to draw a conclusion). Modifying rule weights helps avoid directly modifying the antecedent fuzzy sets, which would otherwise not only adversely affect the prescribed meaning of the associated fuzzy labels, but also require learning a number of parameters for each membership function. The adjustment of rule weights is far less complicated since there is only one single parameter (namely, the weight itself) per rule to learn [13]. In addition to the introduction of methods for rule weight learning [12], [13], [14], [15], the significance and effects of rule weights for fuzzy models have also been discussed in the literature [9], [16].

Inspired by the above observation, this paper proposes a method to learn weighted fuzzy rules with target applications to the development of IDSs. The work takes a two-step approach: A preliminary fuzzy rule base is first generated on the basis of a given quantity space, with initial rule weights heuristically specified, and then, a general-purpose Particle Swarm Optimisation algorithm is devised to simultaneously optimise the rule weights for the acquisition of a set of optimally weighted fuzzy rules. The proposed work is evaluated on benchmark IDS datasets, in comparison with the use of a number of popular machine learning methods.

The reminder of this paper is organised as follows. Section II introduces benchmark IDS data sets, as well as the basics of a fuzzy rule-based system and its relevance to IDSs. Section III describes the proposed methodology of generating a set of optimal weighted fuzzy rules for an IDS. Section IV presents

and discusses comparative experimental results. Section V concludes the paper and outlines ideas for further development.

## II. Background

### A. Benchmark IDS Data

Perhaps, the KDD Cup 99 data set has been the most widely used benchmark in the field of IDS. It is based on the data from DARPA 98, which has been criticised in [17], due to its synthetic characteristics. Nevertheless, KDD Cup 99, which consists of almost 5 million instances, inherited certain problems from its origin, including issues such as the existence of a large number of redundant records [18]. As a result, a new data set, NSL-KDD [18], has been proposed, which is a subset of the records selected from the complete KDD Cup 99 data and which has been studied in a number of recent publications [3], [19].

NSL-KDD also provides an even simpler data set for quicker assessment and validation on the performance of a certain technique, named NSL-KDD-TRN-20, consisting of $20\%$ of the complete NSL-KDD training set. This is adopted as the training set in this work. In the relevant literature, instead of obtaining the testing instances by partitioning a given dataset with cross validation, which is often the case in classical machine learning (especially when the data provided is limited), it is proposed to utilise an independent test set, named NSL-KDD-TST, to objectively evaluate the performance of the proposed approach. Both NSL-KDD-TRN-20 and NSL-KDD-TST are readily available from [18], with NSL-KDD-TRN-20 consisting of $25192$ instances, and NSL-KDD-TST consisting of $22543$ instances.

In terms of attack types, the following four categories [18] are considered here, as opposed to simply discriminating between normal and malicious activities:

- **Denial of Service Attack (DoS)**: is an attack in which the attacker makes certain computing and/or memory resource too busy or too full to handle legitimate requests, denying legitimate users from gaining access to required resources.
- **User to Root Attack (U2R)**: is a class of exploitation in which the attacker starts out with access to a normal user account on the system, exploiting certain vulnerability of the network to gain root access to the system.
- **Remote to Local Attack (R2L)**: occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine, thereby exploiting certain vulnerability to gain local access as a user of that machine.
- **Probing Attack (Probe)**: is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

Generally, the frequency of various attacks is not equally distributed. In particular, the percentage of R2L and U2R is very low compared to the other attacks, due to the fact that they normally only involve one single connection or machine. The distribution of the attacks on both the training and testing data set is summarised in Table I.

TABLE I. Distribution of different attack types

| Class | NSL-KDD-TRN-20 | NSL-KDD-TST |
|---|---|---|
| DoS | 9234 | 7456 |
| U2R | 11 | 202 |
| R2L | 209 | 2754 |
| Probe | 2289 | 2421 |
| Normal | 13449 | 9710 |
| Total | 25192 | 22543 |

### B. Fuzzy Systems and Related Work

The task of learning from or generalising a given problem description, by the use of fuzzy logic and fuzzy sets, is to find a finite set of fuzzy if-then rules capable of reproducing the input-output behaviour of a given system (or process). Without losing generality, the system to be learnt is herein assumed to be a multiple-input-single-output, containing $n$ inputs and one output and involving $m$ patterns for an $M$-class problem. A fuzzy if-then rule $R_j, j = 1, 2, ..., N$, for such a system is represented as follows:

$$\text{If } x_1 \text{ is } A_{j1} \text{ and ... and } x_n \text{ is } A_{jn} \text{ then class is } C_h \text{ with } w_j \tag{1}$$

where $x_1, x_2, ..., x_n$ are the underlying linguistic variables, jointly defining an $n$-dimensional pattern space (with $N$ denoting the number of such fuzzy rules); $A_{ji}$, $i \in \{1, 2, ..., n\}$, is the fuzzy value of the corresponding antecedent $x_i$; $C_h$, $h \in \{1, 2, ..., M\}$, is the consequent class for the $M$-class problem; and $w_j$ is the rule weight of fuzzy rule $R_j$ indicating the strength that any input pattern $X_p = [x_{p1}, x_{p2}, ..., x_{pn}]$, $p \in \{1, 2, ..., m\}$ within the fuzzy subspace delimited by the given antecedent values is deemed to belong to the consequent class $C_h$.

A popular and easy to understand, and perhaps also the simplest method for classifying a new pattern is based on the strategy of "single winner rule" or "winner taking all" [20]. This is employed in this work and the class $C_{X_p}$ of a new pattern $X_p$ is determined by

$$C_{X_p} = \underset{C_h, h=1,2,...M}{\arg\max} \alpha_{C_h} \tag{2}$$

where

$$\alpha_{C_h} = \max\{(\prod_{i=1}^{n} \mu_{A_{ji}}(x_{pi}))w_j | w_j \text{ is associated with } R_j, \\ R_j \text{ is associated with } C_h, j = 1, 2, ..., N\} \tag{3}$$

The inferred class is the consequent of the fuzzy rule that has the maximum value of the antecedent matching degree multiplied by the corresponding rule weight. If two or more classes take the maximum value in Eq. (2) or the matching degree is zero at $X_p$, then the pattern cannot be uniquely classified. To force a classification (if desired), such a pattern may be assigned with a default class label that is associated with most training instances.

In literature, a number of fuzzy rule-based approaches [3], [4], [21], [22] have been proposed to tackle intrusion detection

problems. For example, a genetic fuzzy system [4] within a pairwise learning framework is developed for signature-based IDS. The method [22] designed for anomaly-based IDS works by first generating different training subsets via fuzzy clustering, which are then subsequently generalised by the use of an artificial neural network. A dynamic IDS integrated with the framework of Snort [21], being one of most popular open source IDSs, is designed on the basis of a dynamic version of fuzzy rule interpolation [23].

## III. Methodology

### A. Feature Selection and Fuzzification

Many different features can be monitored by an IDS for the analysis of networking packets. However, the inclusion of all features available does not necessarily help classify networking attacks, as certain features may be either of no relevance to intrusion detection or simply redundant. In the literature, feature selection techniques have been utilised to obtain and use just the more significant attributes without affecting their semantics, but this is beyond the scope of this paper. For simplicity, this work assumes the following four features are returned by a certain feature selection process, which have also been utilised in [3], [24]:

- **src_bytes**: number of data bytes sent by source IP host.
- **dst_bytes**: number of data bytes sent by destination IP host.
- **count**: number of connections to the same host as the current connection in the past 2 seconds.
- **dst_host_diff_rate**: percentage of connections whose ports are different within the past 100 connections with the same destination IP.

These selected features are statistically summarised in Table II. Obviously, they take values from very different ranges. As such, a feature (e.g., src_bytes) with a much broader range of potential values may be dominating the final detection result. To offset this adverse effect, normalisation over these features is required so that the range of all individual features becomes the same, namely within $[0, 1]$. That is, for any feature $x'$ its normalised version $x$ is used in the subsequent analysis such that

$$x = \frac{x' - min(x')}{max(x') - min(x')} \tag{4}$$

TABLE II. Statistics on selected features

| Feature | Minimum | Maximum | Mean | Standard Deviation |
|---|---|---|---|---|
| src_bytes | 0 | 381709090 | 24330.628 | 2410805.402 |
| dst_bytes | 0 | 5151385 | 3491.847 | 88830.718 |
| count | 1 | 511 | 84.591 | 114.673 |
| dst_host_diff_rate | 0 | 1 | 0.083 | 0.187 |

To generate a fuzzy if-then rule base, each dimension of the pattern space is typically divided into $K$ ($K \geq 2$) subsets $\{A_1^K, A_2^K, ..., A_K^K\}$. Practically speaking, partitioning the input space and defining the corresponding fuzzy sets are usually done by domain experts (even though such specification may reflect a certain biased view of particular individuals). In many cases (e.g., [12], [25], [26]), simple fuzzy grid partition of the input space is adopted in order to simplify the generation process. Of course, the performance of a resulting learnt classifier may vary in relation to the variation of the partition of the input space, especially regarding the number of the partitions made.

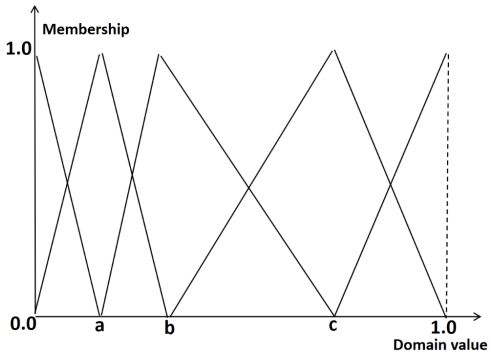


Fig. 1. Membership functions

For illustration, each dimension is herein divided into 5 overlapping triangular fuzzy regions as shown in Fig. 1, where the core of a preceding triangular membership function overlaps with the left support bound of the next adjacent triangular membership function (e.g., point 'a' is core of the second and left support bound of the third membership function in Fig.1). In the absence of expert's knowledge in this work, a data-driven partitioning of individual feature space is performed to reflect the distribution of the underlying data. In particular, fuzzy $c$-means [27] is employed to implement the fuzzification process by clustering each feature space into 3 clusters. Other than the two delimiting values, i.e., 0.0 and 1.0 after the normalisation process (which are defined as rectangular triangular fuzzy sets), three critical points (i.e., a, b and c) are assigned with the three returned cluster centres, respectively. The resulting triangles are taken as predefined fuzzy sets which may be artificially associated with linguistic labels.

### B. Generation of Initial Fuzzy Rule Base

Once predefined fuzzy sets for each individual feature are obtained, a set of preliminary weighted fuzzy rules can be generated by a heuristic approach. In order to generate rules with a variable length for more flexibility without affecting the general representational form, an additional fuzzy set of "Don't care" is added for each attribute, which always returns a full membership value. Given that 4 significant features are utilised in this work with each one partitioned into 5 triangular fuzzy sets, the complete combination of fuzzy sets forming the potential rule antecedent is $(1+5)^4 - 1$. Thus, the identification of each rule consequent class $C_h, h \in \{1, 2, ..., M\}$ of fuzzy rule $R_j$ and the corresponding rule weight $w_j$ can be performed mimicking the method of [16], [26] as follows.

1) Calculate the matching degree for each class $C_h$ with respect to the possible antecedents such that

$$\beta_{C_h} = \sum_{X_p \in C_h} \prod_{i=1}^{n} \mu_{A_{ji}}(x_{pi}) \tag{5}$$

where $X_p$ are the training patterns defined with the corresponding $n$-dimensional fuzzy subspace $A_j = A_{j1} \times A_{j2} \times \cdots \times A_{jn}$, and $\mu_{A_{ji}}(\cdot)$ is the membership function of the antecedent fuzzy set $A_{ji}$.

2) Find $\beta_{C_T}, T = 1, 2, ..., M$, such that

$$\beta_{C_T} = max\{\beta_{C_1}, \beta_{C_2}, ...\beta_{C_M}\} \tag{6}$$

where $C_T$ is the class of the maximum matching degree in response to the antecedent fuzzy sets, forming a candidate if-then rule relating the antecedents and the class.

3) Set the rule weight $w_j$ to each such candidate rule with the following value if its class $C_T$ is the unique one that takes the maximum matching degree in Eq. (6):

$$w_j = (\beta_{C_T} - \beta) / \sum_{h=1}^{M} \beta_{C_h} \tag{7}$$

$$\beta = \sum_{C_h \neq C_T} \beta_{C_h} / (M - 1) \tag{8}$$

where $\beta$ is the sum of the matching degrees for all training patterns belonging to the same fuzzy subspace, except those covered by $C_T$. Otherwise, discard the corresponding candidate rule when two or more classes take the maximum value in Eq. (6) or all the $\beta_{C_T}$ are zero, since it cannot be uniquely determined or there is no training pattern in support of this rule.

4) Promote all remaining candidate rules as the members of the learnt rule base, with their corresponding rule weights assigned, if weights are better than the random guess, i.e., $w_j > \frac{1}{M}$.

Note that the above method for rule generation and rule weight specification is straightforward when a two-class problem is considered. For instance, assuming that $\beta_{C_1} > \beta_{C_2}$, the consequent class is determined to be Class 1 and its weight will be $(\beta_{C_1} - \beta_{C_2})/(\beta_{C_1} + \beta_{C_2})$. Interestingly, suppose that there are almost no Class 2 patterns in the training data set, the result will be $\beta_{C_1} >> \beta_{C_2} \approx 0$ and $w_j \approx 1$. If however, the total matching degrees of patterns for Class 1 and Class 2 are very similar to each other $\beta_{C_1} \approx \beta_{C_2}$, then $w_j \approx 0$.

### *C. Optimisation of Weighted Fuzzy Rules with Particle Swarm Optimisation*

The closer the value of a rule weight is to 1, the more reliable or more significant the rule is. The adjustment of the weights of any two neighbouring rules involves linear computation while determining the corresponding new classification boundary. However, the situation will become much more complicated if the modification of all rule weights is performed simultaneously [12], [26]. That is, whilst the performance of a certain fuzzy rule may be improved by directly changing its rule weight, the performance of its neighbouring fuzzy rules may be deteriorated as a consequence. The overall consequence is thus unpredictable when all the fuzzy rules are changing successively. A method is therefore required to deal with all existing rule weights in a synchronised manner to achieve the overall optimal detection performance.

Broadly speaking, the process of finding an optimal combination of a full set of rule weights appears similar to the behaviour of a particle swarm going towards the best solution with each particle's movement influenced by both its local best position and the currently best known position amongst all rules. This is in principle the same as any typical application of Particle Swarm Optimisation (PSO) [28]. Inspired by this observation and that PSO can encode real numbers directly, PSO is employed to evolve the weights of a fuzzy rule set for the development of an effective IDS. In particular, each of the existing weights is encoded as one particle dimension, and one particle then represents the entire set of the rule weights in the existing fuzzy if-then rules. Positions of the particles in the first generation are initialised with the rule weights obtained by the use of Eqs. (5), (7) and (8). Particles are then iteratively modified towards the best solution with regard to a given quality measure over the set of rule weights.

For each generation, the so-called particle velocity is calculated by the following assignment:

$$v_x = wv_x + c_1 r_1 (x_{gBest} - x) + c_2 r_2 (x_{pBest} - x) \tag{9}$$

where using the terminologies in the literature, $w$ is the inertia weight affecting the trade-off between convergence and exploration-exploitation in the PSO process; $c_1$ and $c_2$ are the social and cognitive scaling parameters, both being a positive constant; $r_1$ and $r_2$ are two random numbers within the range [0, 1]; $x$ is the position for one particle dimension; $x_{gBest}$ is the global best position of all particles, namely the rule weights currently capable of achieving the highest classification accuracy overall; and $x_{pBest}$ is the best individual position where the particular particle $p$ achieves the current best classification accuracy. The position is updated by the assignment: $x = x + \epsilon v_x$, where $\epsilon$ is a real-valued parameter used to control the evolving speed.

In order to measure both the sensitivity of a sub-rule base (its accuracy among instances of the same class, namely, recall r) and the specificity of the subrule base (its accuracy among instances of different classes, namely, precision p), consider the highly imbalanced distribution of network traffic types. The fitness function of each particle is herein gauged by a weighted average of the precision and the recall as follows:

$$F1\_Score = \frac{(1 + \beta^2) \cdot p \cdot r}{(\beta^2 \cdot p + r)} \tag{10}$$

where $\beta = 1$ for F1_Score to evenly weight precision and recall, and the meaning of $p$ and $r$ is described below.

Given that the task of this work is to distinguish the normal traffic from four different types of attacks, the overall precision used is therefore defined an the averaged precision $p$ over $M = 5$ individual classes, such that

$$p = \frac{\sum_{i=1}^{M} p_i}{M} = \frac{\sum_{i=1}^{M} \frac{\mathrm{tp}_i}{\mathrm{tp}_i + \mathrm{fp}_i}}{M} \tag{11}$$

where $p_i$ is the precision, $\mathrm{tp}_i$ and $\mathrm{fp}_i$ are true positive and false positive for type-*i* traffic, respectively. A true positive indicates that the intrusion detection system correctly detects a particular attack having occurred. A false positive indicates that a particular attack has been suggested by the IDS, which

TABLE III. PARAMETER VALUES OF PSO

| $w$ | $c_1$ | $c_2$ | $\epsilon$ | $Max\_Generation$ | $Particle\_Numbers$ |
|---|---|---|---|---|---|
| 1.0 | 2.0 | 2.0 | 1.0 | 200 | 80 |

did not actually occur. Similarly, the overall recall $r$ is defined as:

$$r = \frac{\sum_{i=1}^{M} r_i}{M} = \frac{\sum_{i=1}^{M} \frac{\text{tp}_i}{\text{tp}_i + \text{fn}_i}}{M} \quad (12)$$

where $r_i$ is the recall for type-*i* traffic, and $\text{fn}_i$ is the false negative, indicating that the IDS is unable to detect the intrusion after a particular attack has occurred.

## IV. EXPERIMENTAL RESULTS

### A. Experimental Setup

In order to evaluate the performance of PSO-tunded weighted fuzzy rules (PSO-WRF) for IDS, NSL-KDD-TRN-20 is selected as the training set with NSL-KDD-TST utilised as the independent test set (see Section II-A). Note that as the main aim of this study is to examine the efficacy of PSO-WFR for IDS instead of that of PSO itself, only the basic version of PSO is used in the experiments. The parameter specification for PSO is not carefully adjusted, as given in Table III. Thus, simulation results could be further improved if more sophisticated versions of PSO were used with carefully modified parameters.

To validate the performance of the proposed approach, four popular machine learning algorithms are chosen for comparison. These are: SVM, a sequential optimisation algorithm for building support vector machines with polynomial kernels adopted as the kernel function; KNN, the classical $k$-nearest neighbour approach, where an instance is classified by a majority vote of its neighbours; Naive Bayes, a simple probabilistic learning classifier, based on direct application of the Bayesian theorem with strong independence assumptions; Decision Tree, a classical top down version of C4.5 inductive learning algorithm, by manupilating the concept of information entropy. The implementation of these approaches can be found in WEKA, with default settings [29].

### B. Results and Discussion

As reflected by the training set, the distribution of various classes is highly imbalanced and the numbers of instances for U2R and R2L are particularly low. Therefore, overall accuracy rate may not be sufficient to compare the performance of the designed systems, given the likely biased outcomes. This is also part of the reasons of using F1_Score as the evaluation criterion to tune fuzzy rule weights instead of pure error rates. Tables IV-VIII present the results in terms of precision, recall and F1_Score for each individual traffic class, supported with corresponding complete confusion matrices in Tables X-XIV.

For the type of DoS attack, which consists of approximately 36.65% of overall training instances, all three indicators of the proposed approach are very close to those of SVM, KNN and Decision Tree. Naive Bayes successfully classifies about 91.8% of all DoS attacks from NSL-KDD-TST, yet with only 0.445 precision, making it the worst overall compared to other learning classifiers. Similarly, for Normal traffic, even with a slightly larger proportion of training instances, PSO-WFR also achieves a significantly better F1_Score value over Naive Bayes, and beats the other classifiers with small margins. For the type of Probe attack, PSO-WFR obtains a significantly better precision (0.967) over the rest, albeit the overall F1_Score is not satisfactory given its low coverage on Probe.

TABLE IV. PERFORMANCE COMPARISON ON DOS

| | Precision | Recall | F1_Score |
|---|---|---|---|
| PSO-WFR | 0.816 | 0.686 | 0.745 |
| SVM | 0.841 | 0.692 | 0.760 |
| KNN | 0.837 | 0.684 | 0.753 |
| Naive Bayes | 0.445 | 0.918 | 0.600 |
| Decision Tree | 0.866 | 0.675 | 0.759 |

TABLE V. PERFORMANCE COMPARISON ON U2R

| | Precision | Recall | F1_Score |
|---|---|---|---|
| PSO-WFR | ? | 0.000 | ? |
| SVM | ? | 0.000 | ? |
| KNN | 0.000 | 0.000 | 0.000 |
| Naive Bayes | 0.011 | 0.045 | 0.018 |
| Decision Tree | ? | 0.000 | ? |

TABLE VI. PERFORMANCE COMPARISON ON R2L

| | Precision | Recall | F1_Score |
|---|---|---|---|
| PSO-WFR | 1.000 | 0.034 | 0.066 |
| SVM | ? | 0.000 | ? |
| KNN | 0.477 | 0.015 | 0.030 |
| Naive Bayes | 0.088 | 0.002 | 0.004 |
| Decision Tree | ? | 0.000 | ? |

TABLE VII. PERFORMANCE COMPARISON ON PROBE

| | Precision | Recall | F1_Score |
|---|---|---|---|
| PSO-WFR | 0.967 | 0.240 | 0.385 |
| SVM | 0.707 | 0.454 | 0.553 |
| KNN | 0.665 | 0.463 | 0.546 |
| Naive Bayes | 0.651 | 0.493 | 0.561 |
| Decision Tree | 0.724 | 0.435 | 0.543 |

TABLE VIII. PERFORMANCE COMPARISON ON NORMAL

| | Precision | Recall | F1_Score |
|---|---|---|---|
| PSO-WFR | 0.600 | 0.963 | 0.739 |
| SVM | 0.577 | 0.882 | 0.697 |
| KNN | 0.608 | 0.920 | 0.732 |
| Naive Bayes | 0.884 | 0.407 | 0.557 |
| Decision Tree | 0.591 | 0.930 | 0.723 |

With regard to the type of U2R attack, which only has 11 training instances out of 25192 in total, unfortunately, neither of PSO-WFR, SVM and Decision Tree classifies any of the testing instances as U2R. As the precision is calculated as the number of true positives over the sum of true positives plus false positives, this leads to the result of having a '?'

TABLE IX. OVERALL ACCURACY (%) ON NSL-KDD-TST

| PSO-WFR | SVM | KNN | Naive Bayes | Decision Tree |
|---|---|---|---|---|
| 67.17 | 65.77 | 67.40 | 53.24 | 67.08 |

in Table V for these classifiers, since the denominator of the precision definition becomes 0. According to the confusion matrix for KNN as shown in Table. XI, it incorrectly classifies one observation as U2R. Naive Bayes correctly classifies 9 out of 202 testing instances, but its precision is very low. In terms of R2L attacks, which also occupy a very low proportion (0.830%) of the training instances, SVM and Decision again fail to place any classification on this type of attack just like U2R. However, PSO-WFR has achieved 100% precision, correctly recognising all such R2L attacks. This is meanwhile supported with the largest recall rate than the rest, making PSO-WFR the overall best performer compared to alternative methods.

In general, the overall accuracy of Naive Bayes is much worse than the rest as shown in Table IX. Importantly, the performance of PSO-WFR is in general, better than the powerful SVM and decision tree. Considering the overall F1_Score as the optimisation criterion, PSO-WFR establishes itself as the best performer for detecting R2L attacks, which only constitutes a very small proportion of the training instances. Nevertheless, the overall F1_Score does not work for the U2R type of attack, but this data type only involves 11 out of 25192 training instances and is very difficult to generalise. As such, it remains as a challenging further work to deal with this form of extremely rare attacks.

TABLE X. CONFUSION MATRIX OF SVM

| Class | DoS | U2R | R2L | Probe | Normal | Total |
|---|---|---|---|---|---|---|
| DoS | 5163 | 0 | 0 | 140 | 2153 | 7456 |
| U2R | 0 | 0 | 0 | 7 | 195 | 202 |
| R2L | 1 | 0 | 0 | 110 | 2643 | 2754 |
| Probe | 27 | 0 | 0 | 1100 | 1294 | 2421 |
| Normal | 947 | 0 | 0 | 199 | 8564 | 9710 |
| Total | 6138 | 0 | 0 | 1556 | 14849 | 22543 |

TABLE XI. CONFUSION MATRIX OF IBK

| Class | DoS | U2R | R2L | Probe | Normal | Total |
|---|---|---|---|---|---|---|
| DoS | 5103 | 0 | 26 | 212 | 2115 | 7456 |
| U2R | 14 | 0 | 0 | 5 | 183 | 202 |
| R2L | 162 | 0 | 42 | 117 | 2433 | 2754 |
| Probe | 285 | 0 | 0 | 1120 | 1016 | 2421 |
| Normal | 531 | 1 | 20 | 229 | 8929 | 9710 |
| Total | 6095 | 1 | 88 | 1683 | 14676 | 22543 |

TABLE XII. CONFUSION MATRIX OF NAIVE BAYES

| Class | DoS | U2R | R2L | Probe | Normal | Total |
|---|---|---|---|---|---|---|
| DoS | 6845 | 2 | 0 | 225 | 384 | 7456 |
| U2R | 149 | 9 | 3 | 6 | 35 | 202 |
| R2L | 2497 | 18 | 6 | 166 | 67 | 2754 |
| Probe | 1194 | 0 | 0 | 1193 | 34 | 2421 |
| Normal | 4689 | 770 | 59 | 243 | 3949 | 9710 |
| Total | 15374 | 799 | 68 | 1833 | 4469 | 22543 |

TABLE XIII. CONFUSION MATRIX OF DECISION TREE

| Class | DoS | U2R | R2L | Probe | Normal | Total |
|---|---|---|---|---|---|---|
| DoS | 5033 | 0 | 0 | 164 | 2259 | 7456 |
| U2R | 0 | 0 | 0 | 4 | 198 | 202 |
| R2L | 154 | 0 | 0 | 43 | 2557 | 2754 |
| Probe | 142 | 0 | 0 | 1053 | 1226 | 2421 |
| Normal | 484 | 0 | 0 | 191 | 9035 | 9710 |
| Total | 5813 | 0 | 0 | 1455 | 15275 | 22543 |

TABLE XIV. CONFUSION MATRIX OF PSO-WFR

| Class | DoS | U2R | R2L | Probe | Normal | Total |
|---|---|---|---|---|---|---|
| DoS | 5118 | 0 | 0 | 8 | 2330 | 7456 |
| U2R | 16 | 0 | 0 | 1 | 185 | 202 |
| R2L | 306 | 0 | 93 | 8 | 2347 | 2754 |
| Probe | 477 | 0 | 0 | 580 | 1364 | 2421 |
| Normal | 356 | 0 | 0 | 3 | 9351 | 9710 |
| Total | 6273 | 0 | 93 | 600 | 15577 | 22543 |

## V. CONCLUSION

This paper has proposed an approach to learn an optimal set of weighted fuzzy rules for the purpose of developing effective network intrusion detection systems. As an initial implementation, the proposed work utilises four significant features, whose fuzzy partitions are generated using fuzzy c-means clustering. In this work, a preliminary set of fuzzy rules is created by a heuristic method with initial rule weights specified. This forms the input to a Particle Swarm Optimisation algorithm, such that all existing rule weights are simultaneously optimised. Comparative experimental investigations have been carried out to examine the effectiveness using standard IDS benchmarks. As an important further piece of research, it would be interesting to look into those attacks that are much less frequently seen as the ones that can be dealt with using the current approach.

## REFERENCES


[1] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015.

[3] L. Yang, J. Li, G. Fehringer, P. Barraclough, G. Sexton, and Y. Cao, "Intrusion detection system by fuzzy interpolation," in *Fuzzy Systems, 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[4] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems," *Expert Systems with Applications*, vol. 42, no. 1, pp. 193–202, 2015.

[5] T. Chen, C. Shang, P. Su, and Q. Shen, "Induction of accurate and interpretable fuzzy rules from preliminary crisp representation," *Knowledge-Based Systems*, vol. 146, pp. 152–166, 2018.

[6] P. Su, C. Shang, T. Chen, and Q. Shen, "Exploiting data reliability and fuzzy clustering for journal ranking," *Fuzzy Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 1306–1319, 2017.

[7] T. Chen, P. Su, C. Shang, and Q. Shen, "Reliability-guided fuzzy classifier ensemble," in *Fuzzy Systems, 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[8] P. Su, Q. Shen, T. Chen, and C. Shang, "Ordered weighted aggregation of fuzzy similarity relations and its application to detecting water treatment plant malfunction," *Engineering Applications of Artificial Intelligence*, vol. 66, pp. 17–29, 2017.

[9] H. Ishibuchi and T. Nakashima, "Effect of rule weights in fuzzy rule-based classification systems," *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 506–515, 2001.

[10] T. Chen, Q. Shen, P. Su, and C. Shang, "Induction of quantified fuzzy rules with particle swarm optimisation," in *Fuzzy Systems, 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–7.

[11] S.-M. Zhou and J. Q. Gan, "Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling," *Fuzzy Sets and Systems*, vol. 159, no. 23, pp. 3091–3131, 2008.

[12] T. Chen, Q. Shen, P. Su, and C. Shang, "Fuzzy rule weight modification with particle swarm optimisation," *Soft Computing*, vol. 20, no. 8, pp. 2923–2937, 2016.

[13] M. Z. Jahromi and M. Taheri, "A proposed method for learning rule weights in fuzzy rule-based classification systems," *Fuzzy Sets and Systems*, vol. 159, no. 4, pp. 449–459, 2008.

[14] E. G. Mansoori, M. J. Zolghadri, and S. D. Katebi, "A weighting function for improving fuzzy classification systems performance," *Fuzzy Sets and Systems*, vol. 158, no. 5, pp. 583–591, 2007.

[15] T. Chen, Q. Shen, P. Su, and C. Shang, "Refinement of fuzzy rule weights with particle swarm optimisation," in *Computational Intelligence, 2014 14th UK Workshop on*. IEEE, 2014, pp. 1–7.

[16] H. Ishibuchi and T. Yamamoto, "Rule weight specification in fuzzy rule-based classification systems," *Fuzzy Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 428–435, 2005.

[17] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–6.

[19] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1690–1700, 2014.

[20] H. Ishibuchi, T. Murata, and I. Türkşen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets and Systems*, vol. 89, no. 2, pp. 135–150, 1997.

[21] N. Naik, R. Diao, and Q. Shen, "Dynamic fuzzy rule interpolation and its application to intrusion detection," *IEEE Transactions on Fuzzy Systems*, 2017.

[22] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6225–6232, 2010.

[23] Z. Huang and Q. Shen, "Fuzzy interpolation and extrapolation: A practical approach," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 13–28, 2008.

[24] S. Guha, S. S. Yau, and A. B. Buduru, "Attack detection in cloud infrastructures using artificial neural network with genetic feature selection," in *2016 IEEE International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 2016, pp. 414–419.

[25] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets and Systems*, vol. 52, no. 1, pp. 21–32, 1992.

[26] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms," *Fuzzy Sets and Systems*, vol. 65, no. 2, pp. 237–253, 1994.

[27] R. Suganya and R. Shanthi, "Fuzzy c-means algorithm-a review," *International Journal of Scientific and Research Publications*, vol. 2, no. 11, p. 1, 2012.

[28] J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, no. 2. Perth, Australia, 1995, pp. 1942–1948.

[29] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.